

程序设计基础（C 语言）教程

主 编 刘媛媛 雷莉霞 胡 平
副主编 刘美香 甘 岚

西南交通大学出版社

· 成 都 ·

图书在版编目 (C I P) 数据

程序设计基础 (C 语言) 教程 / 刘媛媛, 雷莉霞, 胡平
主编. — 成都: 西南交通大学出版社, 2022.12

ISBN 978-7-5643-9138-6

I. ①程… II. ①刘… ②雷… ③胡… III. ①C 语言
— 程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字 (2022) 第 255248 号

Chengxu Sheji Jichu (C Yuyan) Jiaocheng 程序设计基础 (C 语言) 教程

主编 刘媛媛 雷莉霞 胡平

责任编辑 黄淑文

封面设计 原谋书装

出版发行 西南交通大学出版社
(四川省成都市金牛区二环路北一段 111 号
西南交通大学创新大厦 21 楼)

发行部电话 028-87600564 028-87600533

邮政编码 610031

网址 <http://www.xnjdcbs.com>

印刷 成都蜀通印务有限责任公司

成品尺寸 185 mm × 260 mm

印张 20.25

字数 503 千

版次 2022 年 12 月第 1 版

印次 2022 年 12 月第 1 次

书号 ISBN 978-7-5643-9138-6

定价 58.00 元

课件咨询电话: 028-87600533

图书如有印装质量问题 本社负责退换

版权所有 盗版必究 举报电话: 028-87600562

前 言

C 语言程序设计是一门基础课程,旨在培养学生具有设计计算机程序、编写程序和调试程序的能力。C 语言是一种通用的高级程序设计语言,同时又具有其他高级语言所不具备的低级语言功能,不但可用于编写应用程序,还可用于编写系统程序,具有运算符和数据类型丰富、生成目标代码质量高、程序执行效率高、可移植性好等特点,因而得到广泛的应用。同时,掌握了 C 语言,就可以较为轻松地学习其他任何一种程序设计语言,为后续的课程打下坚实的基础,能较好地训练学生解决问题的逻辑思维能力以及编程思路 and 技巧,使学生具有较强的利用 C 语言编写软件的能力,为培养学生有较强软件开发能力打下良好基础。

本书针对程序设计思想零基础的同学,以培养学生的计算机思维为目的,从模块化程序设计思想着手,以解决实际问题为课程目标,精心设计了趣味性和实用性较强的案例,由浅入深地介绍了每章所涉及的知识点。全书共分为 10 章,第 1 章通过一个简单的 C 语言程序的概述,介绍了程序的概念和程序设计的流程;第 2 章详细介绍了 C 语言的基础知识;第 3~5 章介绍了 C 程序设计的基本结构;第 6 章介绍了数组,包括一维数组、二维数组、字符数组与字符串;第 7 章介绍了指针;第 8 章介绍了模块化程序设计,包括函数和编译预处理;第 9 章介绍了构造型数据类型的使用;第 10 章介绍了文件系统。

本书可以作为高等院校 C 语言程序设计课程的教材,也可以作为读者自学 C 语言的自学用书。本书由刘媛媛、雷莉霞、胡平担任主编,刘美香、甘岚担任副主编。具体编写分工如下:第 1 章和第 3 章由甘岚编写,第 2 章、第 7 章以及附录由刘媛媛编写,第 6 章和 8 章由雷莉霞编写,第 4 章和 5 章由胡平编写,第 9 章和 10 章由刘美香编写。全书由刘媛媛负责最终统稿。在制订编写大纲及书稿编写过程中,华东交通大学信息工程学院自始至终给予了极大的关心和支持,计算机基础教学部的熊李艳、吴昊、丁振凡、宋岚、周美玲、李明翠、张月圆给了作者大力帮助,在此表示由衷的感谢。

本书不仅有配套的实践教材,而且有电子教案、习题答案等教材中涉及的相关教学资源。由于编者水平有限,编写时间仓促,书中难免有欠妥之处,恳请广大读者提出宝贵意见。

编 者

2022 年 12 月于南昌

目 录

第 1 章 C 语言程序设计概述	1
1.1 程序与程序设计语言	1
1.2 算法及其描述	5
1.3 C 语言的发展及特点	13
1.4 简单 C 语言程序	16
1.5 C 语言程序的执行	19
1.6 小 结	21
1.7 习 题	21
第 2 章 C 语言的基础知识	23
2.1 数据的机内表示	23
2.2 C 语言的基本数据类型	27
2.3 常量和变量	31
2.4 运算符和表达式	36
2.5 运算符的优先级及结合性	44
2.6 表达式的书写规则	45
2.7 各种数据类型的转换	46
2.8 程序举例	49
2.9 小 结	51
2.10 常见的错误	51
2.11 习 题	52
第 3 章 程序设计基本结构——顺序结构	56
3.1 C 语句的描述	56
3.2 数据输入/输出	57
3.3 较复杂的输入输出格式控制	61
3.4 程序举例	67
3.5 小 结	70
3.6 本章常见的编程错误	70
3.7 习 题	71
第 4 章 选择结构	76
4.1 用条件表达式实现选择结构	76
4.2 if 语句	79
4.3 switch 语句	91

4.4	程序举例	94
4.5	小结	97
4.6	本章常见的编程错误	98
4.7	习题	99
第 5 章	循环结构	103
5.1	while 语句	103
5.2	do-while 语句	106
5.3	for 语句	107
5.4	break 和 continue 语句	111
5.5	三种循环结构的比较	113
5.6	循环的嵌套	113
5.7	程序举例	115
5.8	小结	121
5.9	本章常见的编程错误	122
5.10	习题	123
第 6 章	数组	128
6.1	数组的基本概念	128
6.2	一维数组的定义和使用	128
6.3	二维数组的定义和使用	134
6.4	字符数组	140
6.5	数组的应用举例	148
6.6	小结	156
6.7	本章常见的编程错误	157
6.8	习题	158
第 7 章	指针	165
7.1	指针的基本概念	165
7.2	指针运算	170
7.3	指针与数组	172
7.4	程序举例	185
7.5	小结	190
7.6	本章常见的编程错误	190
7.7	习题	191
第 8 章	模块化程序设计	196
8.1	函数的基本概念	196
8.2	函数的定义与声明	199
8.3	函数的参数与返回值	201
8.4	函数的调用	203

8.5	函数的嵌套调用和递归调用	204
8.6	数组作为函数的参数	209
8.7	指针作为函数的参数	212
8.8	函数的返回值为指针	214
8.9	main 函数的参数	215
8.10	变量的作用域与存储类别	216
8.11	编译预处理	226
8.12	程序举例	234
8.13	小 结	237
8.14	本章常见的编程错误	237
8.15	习 题	239
第 9 章	构造型数据类型	245
9.1	结构体型	245
9.2	结构体数组	251
9.3	结构体指针	255
9.4	链 表	258
9.5	共用体	265
9.6	枚举型	269
9.7	程序举例	271
9.8	小 结	273
9.9	本章常见的编程错误	274
9.10	习 题	274
第 10 章	文 件	283
10.1	文件的相关概念	283
10.2	文件的相关操作	286
10.3	小 结	304
10.4	本章常见的编程错误	304
10.5	习 题	304
附录 C	C 语言常用的库函数	307
参考文献	315

第 1 章 C 语言程序设计概述

C 语言从诞生起，就成为主流的程序设计语言，近几年也一直稳居在编程语言排行榜的前列。C 语言是一种计算机程序设计语言，既具有高级语言的特点，又具有汇编语言的特点。因此它可以作为工作系统设计语言，编写系统应用程序，也可以作为应用程序设计语言，编写不依赖计算机硬件的应用程序。它的应用范围广泛，具备很强的数据处理能力，不仅仅是在软件开发上，而且各类科研都需要用到 C 语言。

本章从介绍程序设计语言的基本概念入手，着重介绍 C 语言的发展与特点、应用领域、C 程序的构成及其运行步骤与运行环境。

1.1 程序与程序设计语言

计算机主要由硬件和软件两大部分构成，硬件就不用解释了，主机、显示器等都属于硬件，但是计算机光有硬件是没有办法使用的，还必须有软件支持。软件又分为系统软件，也就是经常用的操作系统，如 Windows XP，Windows 7，Windows 11 等，以及通用软件和应用软件，如 Office 办公软件与 QQ 等，而软件的主体就是程序，因此程序设计语言是计算机科学技术中非常重要的一个部分。

1.1.1 程序设计语言的发展与分类

程序设计语言（Program Design Language，简称 PDL）又称编程语言，是一组用来定义计算机程序的语法规则。它是一种被标准化的交流技巧，用来向计算机发出指令。一种计算机语言让程序员能够准确地定义计算机所需要使用的数据，并精确地定义在不同情况下所应当采取的行动。

正如人们交流思想需要使用各种自然语言（如汉语、英语、法语等）一样，人与计算机之间交流信息必须使用人和计算机都能理解的程序设计语言。程序设计语言也叫计算机语言，是一套关键字和语法规则的集合，可用来产生由计算机进行处理和执行的指令。计算机语言也有一个发展过程，从最开始的计算机语言，也就是二进制机器语言如 011010111，那个时候程序员编程恐怕是非常痛苦的事，因为你要会用 0 和 1 表示一切。后来逐步发展，把一些常用的指令用英语单词表示出来，形成了汇编语言，这个时候也是比较痛苦的，你要记住那些单词的含义不说，还必须知道计算机硬件的组成，再告诉计算机每一步要怎么做，而计算机又是一个非常笨的东西，只要掉一个步骤它就罢工，由于每台机器的硬件组成不一定相同，所以汇编语言的可移植性差，也就是说你在这台计算机上写的程序到另一台计算机上可能就不能用了。之后为了方便软件移植，高级语言诞生了。高级语言不要求程序员掌握计算机的硬件运行，只要写好上层代码，编译软件会将高级语言翻译成汇编语言，然后再将汇编语言转化成计算机语言，从而在计算机中执行。因此，程序员使用高级语言写的代码可以移植到

其他计算机执行，而不用考虑计算机硬件的组成。

程序设计语言有很多种，常用的不过十多种，按照程序设计语言与计算机硬件的联系程度将其分为三类，即机器语言、汇编语言和高级语言。前两类依赖于计算机硬件，有时统称为低级语言，而高级语言与计算机硬件关系较小。

1. 机器语言

机器语言是用二进制代码表示的计算机能直接识别和执行的一种机器指令的集合。它是计算机的设计者通过计算机的硬件结构赋予计算机的操作功能。机器语言具有灵活、直接执行和速度快等特点。不同型号的计算机其机器语言是不相通的，按照一种计算机的机器指令编制的程序，不能在另一种计算机上执行。例如运行在 IBM PC 机上的机器语言程序不能在 51 单片机上运行。

机器指令由操作码和操作数组成，操作码指出要进行什么样的操作，操作数指出完成该操作的数或它在内存中的地址。

例如，计算 1+2 的机器语言程序如下：

```
10110000 00000001    ; 将 1 存入寄存器 AL 中
00000100 00000010    ; 将 2 与寄存器 AL 中的值相加，结果放在寄存器 AL 中
11110100              ; 停机
```

由此可见，用机器语言编写程序，编程人员必须熟记所用计算机的全部指令代码和代码的含义。编写程序时，程序员必须自己处理每条指令和每一数据的存储分配和输入输出，还得记住编程过程中每步所使用的工作单元处在何种状态。这是一件十分烦琐的工作，编写程序花费的时间往往是实际运行时间的几十倍或几百倍。而且，编出的程序全是些 0 和 1 的指令代码，直观性差，难以记忆，还容易出错。

2. 汇编语言

为了克服机器语言的缺点，人们采用了有助于记忆的符号（称为指令助记符）与符号地址来代替机器指令中的操作码和操作数。指令助记符是一些有意义的英文单词的缩写和符号，如用 ADD（Addition）表示加法，用 SUB（Subtract）表示减法，用 MOV（Move）表示数据的传送等等。而操作数可以直接用十进制数书写，地址码可以用寄存器名、存储单元的符号地址等表示。这种表示计算机指令的语言称为汇编语言。

例如上述计算 1+2 的汇编语言程序如下：

```
MOV AL,1              ; 将 1 存入寄存器 AL 中
ADD AL,2              ; 将 2 与寄存器 AL 中的值相加，结果放在寄存器 AL 中
HLT                   ; 停机
```

由此可见，汇编语言克服了机器语言难读难改的缺点，同时保持了占存储空间小、执行速度快的优点，因此许多系统软件的核心部分仍采用汇编语言编制。但是，汇编语言仍是一种面向机器的语言，每条汇编命令都一一对应于机器指令，而不同的计算机的指令长度、寻址方式、寄存器数目等都不一样，这使得汇编语言通用性差，可读性也差。

3. 高级语言

所谓高级语言就是更接近自然语言、更接近数学语言的程序设计语言。它是面向应用的

计算机语言，与具体的机器无关，其优点是符合人类叙述问题的习惯，而且简单易学。高级语言与计算机的硬件结构及指令系统无关，它有更强的表达能力，可以方便地表示数据的运算和程序的控制结构，能更好地描述各种算法，而且容易学习和掌握。但用高级语言编译生成的程序代码一般比用汇编程序语言设计的程序代码要长，执行的速度也慢。

高级语言并不是特指的某一种具体的语言，而是包括很多编程语言，如目前流行的 Java, C, C++, C#, PASCAL, PYTHON, LISP, PROLOG, FoxPro 等，这些语言的语法、命令格式都不相同。

例如上述计算 $1+2$ 的 BASIC 语言程序如下：

```
A=1+2           ; 将 1 加 2 的结果存入变量 A 中
PRINT A         ; 输出 A 的值
END             ; 程序结束
```

这个程序和我们平时的数学思维是相似的，非常直观易懂且容易记忆。

1.1.2 程序设计方法

1. 程序设计过程

计算机程序设计的过程包括问题定义、算法设计、程序设计以及调试运行。整个开发过程都要编制相应的文档，以便管理。

(1) 问题定义。在计算机能够理解一些抽象的名词并做出一些智能的反应之前，必须要对交给计算机的任务做出定义，并最终翻译成计算机能识别的语言。问题定义的方法很多(对此在软件工程的需求分析中会有更多解释，包括描述方法和工具)，但一般包括三个部分：输入、输出和处理。

(2) 算法设计。问题定义确定了未来程序的输入、输出、处理，但并没有具体说明处理的步骤，而算法则是对解决问题步骤的描述。

(3) 程序设计。问题定义和算法设计已经为程序设计规划好了蓝本，下一步就是用真正的计算机语言表达了。不同的语言写出的程序有时会有较大的差别。

(4) 调试运行。程序编写可以在计算机上进行，也可以在纸张上进行，但最终要让计算机来运行则必须输入到计算机中，并经过调试，以便找出错误，然后才能正确地运行。

(5) 文档。对于微小的程序来说，有没有文档显得并不怎么重要，但对于一个需要有多人合作，并且开发、维护较长时间的软件来说，文档就是至关重要的。文档记录程序设计的算法、实现以及修改的过程，保证程序的可读性和可维护性。程序中的注释就是一种很好的文档。

2. 结构化程序设计方法

在早期由于计算机存储器容量非常小，人们设计程序时首先考虑的问题是如何减少存储器开销，硬件的限制不容许人们考虑如何组织数据与逻辑，为此程序员使用各种技巧和手段编写高效的程序。其中显著的特点是程序中大量使用 GOTO 语句，使得程序结构混乱、可读性差、可维护性差、通用性差。但是，随着大容量存储器的出现及计算机技术的广泛应用，程序编写越来越困难，程序的大小以算术级数递增，而程序的逻辑控制难度则以几何级数递增，人们不得不考虑程序设计的方法。

结构化程序设计是进行以模块功能和处理过程设计为主的详细设计的基本原则，其概念最

早由荷兰科学家 E.W.Dijkstra 提出，它的主要观点是采用自顶向下、逐步求精的程序设计方法；使用三种基本控制结构构造程序，任何程序都可用顺序、选择、重复三种基本控制结构构造。

3. 面向对象程序设计

虽然结构化程序设计方法具有很多的优点，但它仍是一种面向过程的程序设计方法，它把数据和处理数据的过程分离为相互独立的实体。当数据结构改变时，所有相关的处理过程都要进行相应的修改，每一种相对于老问题的新方法都要带来额外的开销，程序的可重用性差。

同时由于图形用户界面的应用，程序运行由顺序运行演变为事件驱动，使得软件使用起来越来越方便，但开发起来却越来越困难，对这种软件的功能很难用过程来描述和实现，使用面向过程的方法来开发和维护都将非常困难。

由于上述缺陷已不能满足现代化软件开发的要求，一种全新的软件开发技术应运而生，这就是面向对象程序设计（Object Oriented Programming, OOP）。面向对象程序设计方法于 20 世纪 60 年代后期首次提出，80 年代开始走向实用。

面向对象程序设计是一种计算机编程架构。OOP 的一条基本原则是计算机程序由单个能够起到子程序作用的单元或对象组合而成。OOP 达到了软件工程的三个主要目标：重用性、灵活性和扩展性。OOP=对象+类+继承+多态+消息，其中核心概念是类和对象。面向对象程序设计方法是尽可能模拟人类的思维方式，使得软件的开发方法与过程尽可能接近人类认识世界、解决现实问题的方法和过程，也使得描述问题的问题空间与问题的解决方案空间在结构上尽可能一致，把客观世界中的实体抽象为问题域中的对象。面向对象程序设计以对象为核心，该方法认为程序由一系列对象组成。

1.1.3 程序设计语言翻译系统

计算机硬件只能识别并执行机器指令，即只能直接执行相应机器语言格式的代码程序，而不能直接执行高级语言或汇编语言编写的程序。为了让计算机能够理解高级语言或汇编语言编写的程序，必须要为它配备一个“翻译”，这就是所谓的程序设计语言翻译程序。

程序设计语言翻译程序是一类系统软件，通常所说的翻译程序是指这样一个程序，它能够把某一种语言程序（称为源语言程序）转换成另一种语言程序（称为目标语言程序），而后者与前者在逻辑上是等价的。不同的程序设计语言需要有不同的程序语言翻译系统，同一种程序设计语言在不同类型的计算机上也需要配置不同的程序语言翻译系统。

程序设计语言翻译系统可以分成 3 种：汇编语言翻译系统、高级语言翻译系统和高级语言解释系统。这些翻译系统之间的不同之处主要体现在它们生成计算机可以执行的机器语言的过程中。

1. 汇编语言翻译系统

汇编语言翻译系统（汇编程序）的主要功能是将汇编语言书写的源程序，翻译成用二进制代码 0 或 1 表示的等价的机器语言，形成计算机可以执行的机器指令代码，如图 1-1 所示。

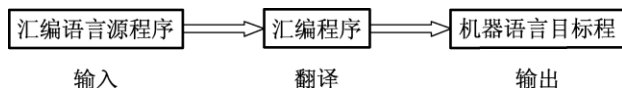


图 1-1 汇编程序功能示意图

2. 高级程序设计语言翻译系统

高级程序设计语言翻译系统是指将用高级语言编写的源程序翻译成等价的汇编语言程序或机器语言程序的处理系统，也称为编译程序。

由此可见，在计算机上用编译方式执行高级语言编写的程序，一般需要两个阶段：第一阶段称为编译阶段，其任务是由编译程序将源程序翻译为目标程序，如果目标程序不是机器语言程序，则尚需汇编程序再行汇编为机器代码程序；第二阶段为运行阶段，其任务是在目标计算机上执行编译阶段所得到的目标程序。编译程序和运行系统合称为编译系统。图 1-2 显示了按编译方式执行一个高级语言的主要步骤。

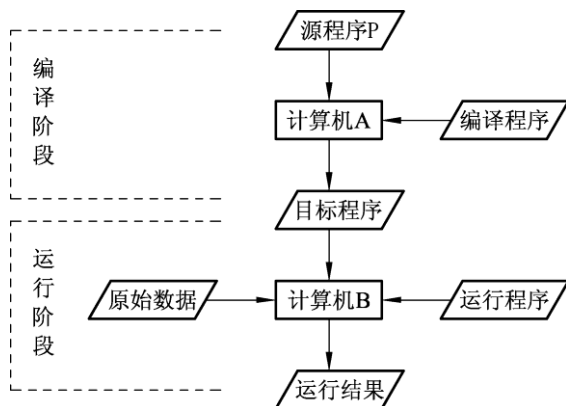


图 1-2 计算机执行高级语言程序的步骤

3. 高级程序设计语言解释系统

高级程序设计语言解释系统是按源程序中的语句的动态顺序逐条翻译并立即执行相应功能的处理系统。它将源语言(如 BASIC)书写的源程序作为输入，解释一句后就提交计算机执行一句，并不形成目标程序。就像外语翻译中的“口译”一样，说一句译一句，不产生全文的翻译文本。这种工作方式非常适合于人通过终端设备与计算机会话，如在终端上打一条命令或语句，解释程序就立即将此语句解释成一条或几条指令并提交硬件立即执行且将执行结果反映到终端，从终端把命令打入后，就能立即得到计算结果。

对源程序边解释翻译成机器代码边执行的高级语言程序，由于它的方便性和交互性较好，早期一些高级语言采用这种方式，如 BASIC。但它的弱点是运行效率低，程序的运行依赖于开发环境，不能直接在操作系统下运行。

1.2 算法及其描述

1.2.1 算法的概念

1. 什么是算法

计算机解题一般可分解成若干操作步骤，通常把完成某一任务的操作步骤称为求解该问题的算法。程序就是用计算机语言描述的算法。

算法是指完成一个任务所需要的具体步骤和方法。也就是说给定初始状态或输入数据，

能够得出所要求或期望的终止状态或输出数据。

既然算法是解决给定问答的方法, 算法所处理的对象就是该问题所涉及的数据。程序的目的是加工数据, 而如何加工数据就是算法的目的。

例如给定两个正整数 m 和 n , 求它们的最大公约数。在学数学的时候, 我们都知道这个问题就是求能同时整除 m 和 n 的最大正整数。但是要在计算机中实现的话, 仅有数学的思维是不行的, 计算机中实现的步骤如下:

- (1) 以 n 除 m 并令所得余数为 r , r 必小于 n ;
- (2) 若 $r=0$ 算法结束, 输出结果 n 。否则继续步骤 (3);
- (3) 将 m 替换为 n , n 替换为 r , 并返回步骤 (1) 继续进行。

2. 算法的性质

著名计算机科学家 Donald Knuth 在他的著作《The Art of Computer Programming》中曾把算法的性质归纳为以下 5 点:

- (1) 输入: 一个算法必须有零个或以上输入量。
- (2) 输出: 一个算法应有一个或以上输出量, 输出量是算法计算的结果。
- (3) 明确性: 算法的描述必须无歧义, 以保证算法的实际执行结果精确地符合要求或期望, 通常要求实际运行结果是确定的。
- (4) 有限性: 依据图灵的定义, 一个算法是能够被任何图灵完备系统模拟的一串运算, 而图灵机只有有限个状态、有限个输入符号和有限个转移函数(指令)。而一些定义更规定算法必须在有限个步骤内完成任务。
- (5) 有效性: 又称可行性。能够实现, 算法中描述的操作都可以通过已经实现的基本运算执行有限次来实现。

1.2.2 算法的描述

算法是对解题过程的精确描述。定义解决问题的算法对程序员来说通常是最具挑战性的任务。它既是一种技能又是一门艺术, 要求程序员懂得程序设计概念并具有创造性。对算法的描述是建立在语言基础之上的。在将算法转化为高级语言源程序之前, 通常先采用文字或图形工具来描述算法。文字工具如自然语言、伪代码等, 图形工具如流程图、N-S 流程图等。

1. 自然语言







自然语言即人们日常生活中所用的语言, 如汉语、英语等。使用自然语言不用专门训练, 所描述的算法也通俗易懂。然而其缺点也是明显的: 首先是由于自然语言的歧义性容易导致算法执行的不确定性; 其次是由于自然语言表示的串行性, 因此当一个算法中循环和分支较多时, 就很难清晰地表示出来; 此外, 自然语言表示的算法不便转换成用计算机程序设计语言表示的程序。

2. 流程图

流程图是采用一些框图符号来描述算法的逻辑结构, 每个框图符号表示不同性质的操作。流程图可以很方便地表示任何程序的逻辑结构。另外, 用流程图表示的算法不依赖于任何具

体的计算机和程序设计语言，从而有利于不同环境的程序设计。早在 20 世纪 60 年代，美国国家标准协会（American National Standards Institute, ANSI）就颁布了流程图的标准，这些标准规定了用来表示程序中各种操作的流程图符号，例如用矩形表示处理，用菱形表示判断，用平行四边形表示输入/输出，用带箭头的折线表示流程，等等。如表 1-1 所示流程图符号及意义。

表 1-1 流程图常用符号

流程图符号	名 称	说 明
	起止框	表示算法的开始和结束
	处理框	表示完成某种操作，如初始化或运算赋值等
	判断框	表示根据一个条件成立与否，决定执行两种不同操作的其中一个
	输入输出框	表示数据的输入输出操作
	流程线	用箭头表示程序执行的流向
	连接点	用于流程分支的连接

3. N-S 流程图

N-S 流程图又称为结构化流程图，于 1973 年由美国学者 I.Nassi 和 B.Shnei-derman 提出。与传统流程图不同的是，N-S 流程图不用带箭头的流程线来表示程序流程的方向，而采用一系列矩形框来表示各种操作，全部算法写在一个大的矩形框内，在大框内还可以包含其他从属于它的小框，这些框一个接一个从上向下排列，程序流程的方向总是从上向下。N-S 结构化流程图比较适合于表达三种基本结构（顺序、选择、循环），适于结构化程序设计，因此很受程序员欢迎。

N-S 流程图用以下不同的流程图符号表示不同的结构：

(1) 顺序结构。顺序结构用图 1-3 形式表示。A 和 B 两个框组成一个顺序结构。

(2) 选择结构。选择结构用图 1-4 表示，它与图 1-3 相应。当 P 条件成立时执行 A 操作，P 不成立则执行 B 操作。注意：图 1-4 是一个整体，代表一个基本结构。

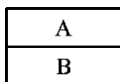


图 1-3 顺序结构

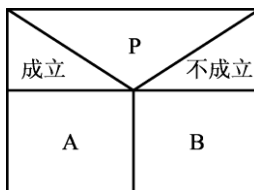


图 1-4 选择结构

(3) 循环结构。“当”型循环结构用图 1-5 形式表示。图 1-5 表示当 P1 条件成立时反复执行 A 操作，直到 P1 条件不成立为止。“直到”型循环结构用图 1-6 形式表示。

在初学时，为清楚起见，可如图 1-5 和图 1-6 那样，写明“当 P1”或“直到 P2”，待熟练之后，可以不写“当”和“直到”字样，只写“P1”和“P2”。从图的形状即可知道是“当”型或“直到”型。

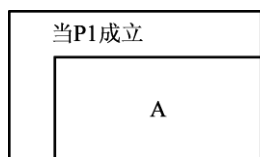


图 1-5 “当”型循环结构

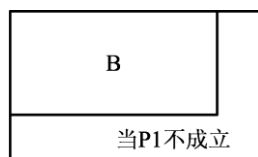


图 1-6 “直到”型循环结构

用以上 3 种 N-S 流程图中的基本框可以组成复杂的 N-S 流程图，以表示算法。

应当说明，在图中的 A 框或 B 框，可以是一个简单的操作（如读入数据或打印输出等），也可以是 3 个基本结构之一。

例如，图 1-7 所表示的顺序结构，其中的 A 框可以又是一个选择结构，B 框可以又是一个循环结构。如图 1-8 所示那样，由 A 和 B 这两个基本结构组成一个顺序结构。

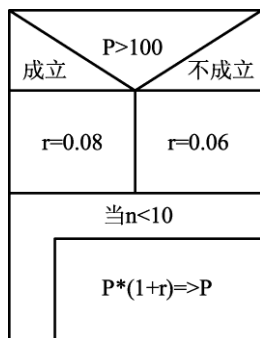


图 1-7 复杂的 N-S 流程图 1

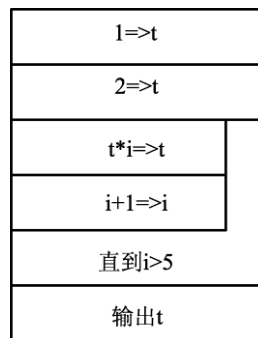


图 1-8 复杂的 N-S 流程图 2

4. 伪代码

伪代码是指不能够直接编译运行的程序代码，它是用介于自然语言和计算机语言之间的文字和符号来描述算法和进行语法结构讲解的一个工具。表面上它很像高级语言的代码，但又不像高级语言那样要接受严格的语法检查。它比真正的程序代码更简明，更贴近自然语言。它不用图形符号，因此书写方便，格式紧凑，易于理解，便于向计算机程序设计语言算法程序过渡。用伪代码书写算法时，既可以采用英文字母或单词，也可以采用汉字，以便于书写和阅读。它没有固定的、严格的语法规则，只要把意思表达清楚即可。用伪代码描述算法时，自上而下地写。每一行（或每几行）表示一个基本操作。用伪代码书写的算法格式紧凑，易于理解，便于转化为计算机语言算法（即程序）。在书写时，伪代码采用缩进格式来表示三种基本结构。一个模块的开始语句和结束语句都靠着左边界书写，模块内的语句向内部缩进一段距离，选择结构和循环结构内的语句再向内缩进一段距离。这样的话，算法书写格式一致，富有层次，清晰易读，能直观地区别出控制结构的开始和结束。

5. 程序设计语言

对一些简单的问题，可以直接使用某种程序设计语言来描述算法。

6. 算法设计举例

【例 1-1】若给定两个正整数 m 和 n ，试写出求它们的最大公约数（即能同时整除 m 和 n 的最大正整数）的算法。

解：在公元前 300 年左右，欧几里得在其著作《几何原本》(Elements) 中阐述了求解两个数最大公约数的过程。

(1) 该算法用自然语言描述如下：

第 1 步：读入两个正整数 m 和 n ，大的数存入 m ，小的数存入 n ；

第 2 步：求 m 除以 n 的余数 r ；

第 3 步：用 n 的值取代 m ， r 的值取代 n ；

第 4 步：判别 r 的值是否为零，如果 $r=0$ ，则算法结束，输出 m 的值，即为最大公因子；否则返回第 2 步。

(2) 流程图如图 1-9 所示。

(3) N-S 流程图如图 1-10 所示。

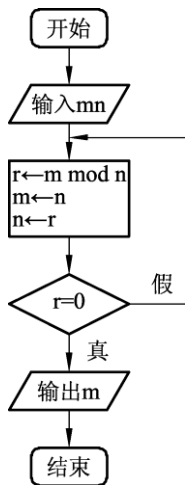


图 1-9 例 1-1 流程图

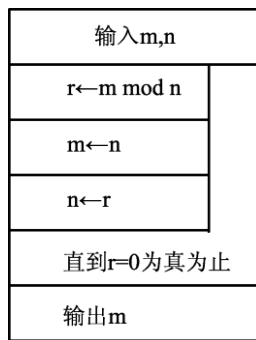


图 1-10 例 1-1 N-S 流程图

(4) 伪代码描述表示如下：

```

INPUT m,n
DO
    r=m MOD n
    m=n
    n=r
LOOP UNTIL r=0
PRINT m
END
  
```

(5) C 语言描述如下:

C 源程序 (文件名: li1_1.c):

```
#include <stdio.h>
int main()
{
    int m,n,r;
    printf("请输入 m,n:");
    scanf("%d%d",&m,&n);
    do
    {
        r = m%n;
        m = n;
        n = r;
    }while (r);
    printf("最大公约数: %d\n",m);
    return 0;
}
```



li1_1.c

【例 1-2】 求 $1 + 2 + 3 + \dots + 100$ 之和。分别用传统流程图、N-S 流程图及自然语言描述其算法, 并将该算法转化为 BASIC 语言源程序。设变量 x 表示被加数, y 表示加数。

解: (1) 该算法用自然语言描述如下:

步骤 1: 将 1 赋值给 x ;

步骤 2: 将 2 赋值给 y ;

步骤 3: 将 x 与 y 相加, 结果存放在 x 中;

步骤 4: 将 y 加 1 结果存放在 y 中;

步骤 5: 若 y 小于或等于 100 转到步骤 3 继续执行, 否则算法结束, 结果为 x 。

(2) 流程图如图 1-11 所示。

(3) N-S 流程图如图 1-12 所示。

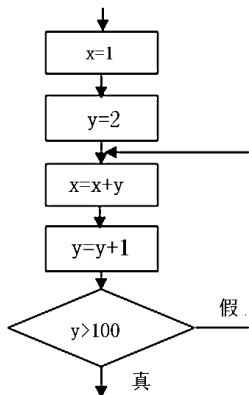


图 1-11 例 1-2 流程图

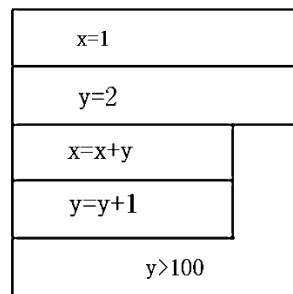


图 1-12 例 1-2 N-S 流程图

(4) 伪代码描述表示如下：

```
x = 1
y = 2
WHILE y <= 100
x = x+y
y = y+1
WEND
print "x = "; x
END
```

(5) C 语言描述如下：

C 源程序（文件名：li1_2.c）：

```
#include <stdio.h>
int main()
{
    int x,y;
    x=1;
    y=2;
    do
    {
        x = x+y;
        y=y+1;
    }while(y<=100);
    printf("1+2+3+...+100=%d\n",x);
}
```



li1_2.c

【例 1-3】描述商家给客户打折问题，规定一种商品一次消费金额超过 200 元的客户可以获得折扣（10%）。

解：（1）该算法用自然语言描述如下：

步骤 1：输入 price，qyt；

步骤 2：price 和 qyt 相乘赋给 sum；

步骤 3：判断 sum 是否大于 200，如果大于转到步骤 4，否则转到步骤 5；

步骤 4：sum 乘以 0.1 赋值给 discount，sum 减去 discount 赋给 rsum 结束；

步骤 5：sum 赋给 rum 结束。

（2）流程图如图 1-13 所示。

（3）N-S 流程图如图 1-14 所示。

（4）伪代码描述表示如下：

```
sum = qyt * price
if ( sum > 200 )
    discount = sum * 0.1
    rsum = sum - discount
```

```
else
```

```
    rsum = sum
```

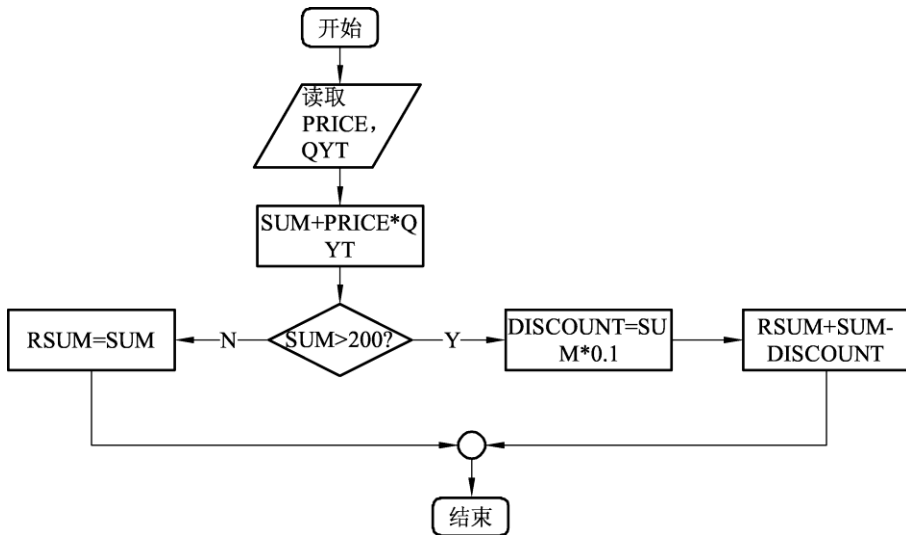


图 1-13 例 1-3 传统流程图

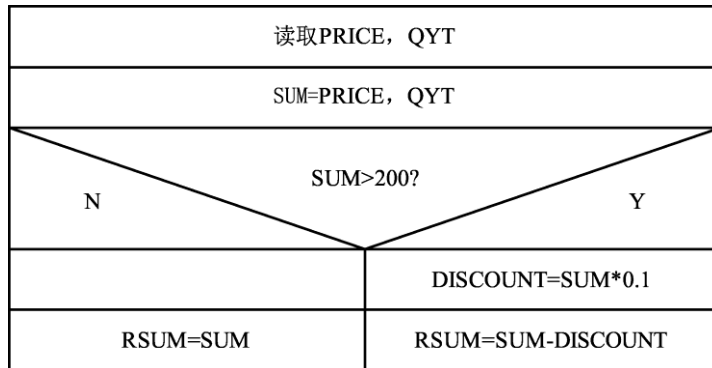


图 1-14 例 1-3 N-S 流程图

(5) C 语言描述如下:

C 源程序 (文件名: li1_3.c):

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int price,qyt,sum,discount,rsum;
```

```
    printf("请输入 price,qyt:");
```

```
    scanf("%d%d",&price, &qyt);
```

```
    sum = price *qyt;
```

```
    if (sum > 200)
```

```
    {
```



li1_3.c

```
        discount = sum * 0.1;
        rsum = sum - discount;
    }
else
{
    rsum = sum;
}
printf("sum=%d,rsum=%d\n",sum,rsum);
}
```

1.3 C 语言的发展及特点

1.3.1 C 语言的发展

C 语言最早的原型是 ALGOL 60。

1963 年，剑桥大学将其发展成为 CPL(Combined Programing Language)。

1967 年，剑桥大学的 Martin Richards 对 CPL 语言进行了简化，产生了 BCPL (Basic Combined Programming Language) 语言。

20 世纪 60 年代，美国 AT&T 公司贝尔实验室 (AT&T Bell Laboratory) 的研究员 Ken Thompson 闲来无事，手痒难耐，想玩一个他自己编的、模拟在太阳系航行的电子游戏——Space Travel。他背着老板，找到了一台空闲的机器——PDP-7。但这台机器没有操作系统，而游戏必须使用操作系统的一些功能，于是他着手为 PDP-7 开发操作系统。后来，这个操作系统被命名为——UNIX。

1970 年，美国贝尔实验室的 Ken Thompson，以 BCPL 语言为基础，设计出很简单且很接近硬件的 B 语言 (取 BCPL 的首字母)，并且他用 B 语言写了第一个 UNIX 操作系统。

1971 年，同样酷爱 Space Travel 的 Dennis M.Ritchie 为了能早点儿玩上游戏，加入了 Thompson 的开发项目，合作开发 UNIX。他的主要工作是改造 B 语言，使其更成熟。

1972 年，美国贝尔实验室的 D.M.Ritchie 在 B 语言的基础上最终设计出了一种新的语言，他取了 BCPL 的第二个字母作为这种语言的名字，这就是 C 语言。

1973 年初，C 语言的主体完成。Thompson 和 Ritchie 迫不及待地开始用它完全重写了 UNIX。此时，编程的乐趣使他们已经完全忘记了那个“Space Travel”，一门心思地投入到了 UNIX 和 C 语言的开发中。随着 UNIX 的发展，C 语言自身也在不断地完善。直到今天，各种版本的 UNIX 内核和周边工具仍然使用 C 语言作为最主要的开发语言，其中还有不少继承 Thompson 和 Ritchie 之手的代码。

在开发中，他们还考虑把 UNIX 移植到其他类型的计算机上使用。C 语言强大的移植性 (Portability) 在此显现。机器语言和汇编语言都不具有移植性，为 x86 开发的程序，不可能在 Alpha, SPARC 和 ARM 等机器上运行。而 C 语言程序则可以在任意机器上运行，只要那种计算机上有 C 语言编译器和库。随后不久，UNIX 的内核(Kernel)和应用程序全部用 C 语言

改写, 从此, C 语言成为 UNIX 环境下使用最广泛的主流编程语言。

1977 年, Dennis M.Ritchie 发表了不依赖于具体机器系统的 C 语言编译文本——《可移植的 C 语言编译程序》。

1978 年, Dennis Ritchie 和 Brian Kernighan 合作推出了《The C Programming Language》的第一版(按照惯例, 经典著作一定有简称, 该著作简称为 K&R), 书末的参考指南(Reference Manual) 一节给出了当时 C 语言的完整定义, 成为那时 C 语言事实上的标准, 人们称之为 K&R C。从这以后, C 语言被移植到了各种机型上并受到了广泛的支持, 使 C 语言在当时的软件开发中几乎一统天下。

随着 C 语言在多个领域的推广、应用, 一些新的特性不断被各种编译器实现并添加进来。于是, 建立一个新的“无歧义、与具体平台无关的 C 语言定义”成为越来越重要的事情。1982 年, 很多有识之士和美国国家标准协会为了使这个语言健康地发展下去, 决定成立 C 标准委员会, 建立 C 语言的标准。委员会由硬件厂商、编译器及其他软件工具生产商、软件设计师、顾问、学术界人士、C 语言作者和应用程序员组成。

1983 年, ASC X3(ANSI 属下专门负责信息技术标准化的机构, 现已改名为 INCITS)成立了一个专门的技术委员会 J11 (J11 是委员会编号, 全称是 X3J11), 负责起草关于 C 语言的标准草案。

1989 年, ANSI 发布了第一个完整的 C 语言标准——ANSI X3.159—1989, 简称“C89”, 不过人们也习惯称其为“ANSIC”。C89 在 1990 年被国际标准组织 ISO (International Organization for Standardization) 一字不改地采纳, 所以也有“C90”的说法。1999 年, 在做了一些必要的修正和完善后, ISO 发布了 C 语言标准, 命名为 ISO/IEC 9899: 1999, 简称“C99”。

随后,《The C Programming Language》第二版开始出版发行, 书中内容根据 ANSI C (C89) 进行了更新。1990 年, 在 ISO/IEC JTC1/SC22/WG14(ISO/IEC 联合技术第 I 委员会第 22 分委员会第 14 工作组)的努力下, ISO 批准 ANSI C 成为国际标准。于是 ISO C (又称为 C90) 诞生了。除了标准文档在印刷编排上的某些细节不同外, ISO C (C90) 和 ANSI C (C89) 在技术上完全一样。

之后, ISO 在 1994、1996 年分别出版了 C90 的技术勘误文档, 更正了一些印刷错误, 并在 1995 年通过了一份 C90 的技术补充, 对 C90 进行了微小的扩充, 经过扩充后的 ISO C 被称为 C95。

1999 年, ANSI 和 ISO 又通过了最新版本的 C 语言标准和技术勘误文档, 该标准被称为 C99。这基本上是目前关于 C 语言的最新、最权威的定义了。

现在, 各种 C 编译器都提供了 C89 (C90) 的完整支持, 对 C99 还只提供了部分支持, 还有一部分提供了对某些 K&R C 风格的支持。

2011 年 12 月 8 日, ISO 正式发布了 C 语言的新标准 C11, 之前被称为 C1X, 官方名称为 ISO/IEC 9899:2011。

2018 年 6 月, ISO 发布了 ISO/IEC 9899:2018 标准, 这个标准被称为 C18, 是目前最新的 C 语言编程标准, 该标准主要是对 C11 进行了补充和修正, 并没有引入新的语言特性。

1.3.2 C 语言的特点

C 语言具有以下优点：

(1) 简洁紧凑、灵活方便。

C 语言一共只有 32 个关键字、9 种控制语句，程序书写形式自由，区分大小写。把高级语言的基本结构和语句与低级语言的实用性结合起来。C 语言可以像汇编语言一样对位、字节和地址进行操作，而这三者是计算机最基本的工作单元。

(2) 运算符丰富。

C 语言的运算符包含的范围很广泛，共有 34 种运算符。C 语言把括号、赋值、强制类型转换等都作为运算符处理。从而使 C 语言的运算类型极其丰富，表达式类型多样化。灵活使用各种运算符可以实现在其他高级语言中难以实现的运算。

(3) 数据类型丰富。

C 语言有整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等数据类型，能用来实现各种复杂的数据结构的运算，并引入了指针概念，使程序效率更高。

(4) 表达方式灵活实用。

C 语言提供多种运算符和表达式值的方法，对问题的表达可通过多种途径获得，其程序设计更主动、灵活。它语法限制不太严格，程序设计自由度大，如对整型量与字符型数据及逻辑型数据可以通用等。

(5) 允许直接访问物理地址，对硬件进行操作。

C 语言允许直接访问物理地址，可以直接对硬件进行操作，因此它既具有高级语言的功能，又具有低级语言的许多功能，能够像汇编语言一样对位 (bit)、字节和地址进行操作，而这三者是计算机最基本的工作单元，可用来写系统软件。

(6) 生成目标代码质量高，程序执行效率高。

C 语言描述问题比汇编语言迅速，工作量小、可读性好，易于调试、修改和移植，而代码质量与汇编语言相当。C 语言一般只比汇编程序生成的目标代码效率低 10% ~ 20%。

(7) 可移植性好。

C 语言在不同机器上的 C 编译程序，86% 的代码是公共的，所以 C 语言的编译程序便于移植。在一个环境上用 C 语言编写的程序，不改动或稍加改动，就可移植到另一个完全不同的环境中运行。

(8) 表达力强。

C 语言有丰富的数据结构和运算符。包含了各种数据结构，如整型、数组类型、指针类型和联合类型等，可用来实现各种数据结构的运算。C 语言的运算符有 34 种，范围很宽，灵活使用各种运算符可以实现难度极大的运算。

C 语言能直接访问硬件的物理地址，能进行位 (bit) 操作。兼有高级语言和低级语言的许多优点。它既可用于编写系统软件，又可用于开发应用软件，已成为一种通用程序设计语言。另外 C 语言具有强大的图形功能，支持多种显示器和驱动器，且计算功能、逻辑判断功能强大。

C 语言也存在一些缺点：

(1) C 语言在数据的封装性上不好，这一点使得 C 在数据的安全性上有很大缺陷，这也

是 C 和 C++ 的一大区别。

(2) C 语言的语法限制不太严格, 对变量的类型约束不严格, 影响程序的安全性, 对数组下标越界不作检查等。

(3) 从应用的角度, C 语言比其他高级语言较难掌握。也就是说, 要求用 C 语言的人对程序设计更熟练一些。

1.3.3 C 语言的应用领域

(1) 应用软件。Linux 操作系统中的应用软件都是使用 C 语言编写的, 因此这样的应用软件安全性非常高。

(2) 对性能要求严格的领域。一般对性能有严格要求的地方都是用 C 语言编写的, 比如网络程序的底层和网络服务器端底层、地图查询等。

(3) 系统软件和图形处理。C 语言具有很强的绘图能力和可移植性, 并且具备很强的数据处理能力, 可以用来编写系统软件、制作动画、绘制二维图形和三维图形等。

(4) 数字计算。相对于其他编程语言, C 语言是数字计算能力超强的高级语言。

(5) 嵌入式设备开发。手机、PAD 等时尚消费类电子产品相信大家都不陌生, 其内部的应用软件、游戏等很多都是采用 C 语言进行嵌入式开发的。

(6) 游戏软件开发。游戏大家更不陌生, 很多人就是由玩游戏而熟悉了计算机。利用 C 语言可以开发很多游戏, 比如推箱子、贪吃蛇等。

1.4 简单 C 语言程序

1.4.1 C 语言实例

为了说明 C 语言源程序结构的特点, 先看以下几个程序。这几个程序由简到难, 表现了 C 语言源程序在组成结构上的特点。虽然有关内容还未介绍, 但可从这些例子中了解到组成一个 C 源程序的基本部分和书写格式。

【例 1-4】 在屏幕上输出 Hello, World!

C 源程序: (文件名: li1_4.c)

```
#include<stdio.h>           /*文件包含*/
Void main()                 /*主函数 */
{                             /*函数体开始*/
    printf("Hello, World! \n"); /*输出语句*/
}                             /*函数体结束*/
```



li1_4.c

说明: main 是主函数名, 每个 C 程序必须有一个主函数 main; void 是函数类型, 表示空类型, 说明主函数 main 没有返回值。{ } 是函数开始和结束的标志, 不可省。每个 C 语句以分号结束。C 语言提供了很多标准库函数供用户使用, 使用时应在程序开头一行写:

#include <stdio.h>。

【例 1-5】 输入一个 x, 求出 sin(x) 并在屏幕上显示。

C 源程序: (文件名: li1_5.c)



li1_5.c


```

#include<math.h>           /* include 称为文件包含命令*/
#include<stdio.h>         /*扩展名为.h 的文件称为头文件 */
Void main()              /* 主函数*/
{
    double x,s;          /*定义两个实数变量，以被后面程序使用*/
    printf("input number:\n"); /*显示提示信息 */
    scanf("%lf",&x);     /*从键盘获得一个实数 x */
    s=sin(x);           /*求 x 的正弦,并把它赋给变量 s*/
    printf("sine of %lf is %lf\n",x,s); /*显示程序运算结果*/
}

```

说明：

程序的功能是从键盘输入一个数 x ，求 x 的正弦值，然后输出结果。在 `main()` 之前的两行称为预处理命令（详见后面）。预处理命令还有其他几种，这里的 `include` 称为文件包含命令，其意义是把尖括号 `<>` 或引号 `" "` 内指定的文件包含到本程序来，成为本程序的一部分。被包含的文件通常是由系统提供的，其扩展名为 `.h`。因此也称为头文件或首部文件。C 语言的头文件中包括了各个标准库函数的函数原型。因此，凡是在程序中调用一个库函数时，都必须包含该函数原型所在的头文件。在本例中，使用了三个库函数：输入函数 `scanf`，正弦函数 `sin`，输出函数 `printf`。`sin` 函数是数学函数，其头文件为 `math.h` 文件，因此在程序的主函数前用 `include` 命令包含了 `math.h`。`scanf` 和 `printf` 是标准输入输出函数，其头文件为 `stdio.h`，在主函数前也用 `include` 命令包含了 `stdio.h` 文件。

需要说明的是，C 语言规定对 `scanf` 和 `printf` 这两个函数可以省去对其头文件的包含命令。所以在本例中也可以删去第二行的包含命令 `#include<stdio.h>`。

同样，在例 1-4 中使用了 `printf` 函数，也省略了包含命令。

在例题中的主函数体中又分为两部分，一部分为说明部分，另一部分为执行部分。说明是指变量的类型说明。例题 1-4 中未使用任何变量，因此无说明部分。

C 语言规定，源程序中所有用到的变量都必须先说明、后使用，否则将会出错。这一点是编译型高级程序设计语言的一个特点，与解释型的 BASIC 语言是不同的。说明部分是 C 源程序结构中很重要的组成部分。本例中使用了两个变量 x 和 s ，用来表示输入的自变量和 `sin` 函数值。由于 `sin` 函数要求这两个量必须是双精度浮点型，故用类型说明符 `double` 来说明这两个变量。说明部分后的 4 行为执行部分或称为执行语句部分，用以完成程序的功能。执行部分的第 1 行是输出语句，调用 `printf` 函数在显示器上输出提示字符串，请操作人员输入自变量 x 的值。第 2 行为输入语句，调用 `scanf` 函数，接受键盘上输入的数并存入变量 x 中。第 3 行是调用 `sin` 函数并把函数值送到变量 s 中。第 4 行是用 `printf` 函数输出变量 s 的值，即 x 的正弦值。程序结束。

运行本程序时，首先在显示器屏幕上给出提示串 `input number`，这是由执行部分的第一行完成的。用户在提示下从键盘上键入某一数，如 5，按下回车键，接着在屏幕上给出计算结果。

【例 1-6】 求 3 个数中较大者。

C 源程序：（文件名：li1_6.c）：

```
#include <stdio.h>
```



li1_6.c

```

void main( )                /* 主函数*/
{
    int max(int x,int y);    /* 对被调用函数 max 的声明 */
    int a, b, c;            /*定义变量 a、b、c */
    scanf("%d,%d",&a,&b) ;  /*输入变量 a 和 b 的值*/
    c=max(a,b) ;           /*调用 max 函数,将得到的值赋给 c */
    printf("max=%d\n",c) ; /*输出 c 的值*/
}
int  max(int x, int y)      /*定义函数 max*/
{
    int z;                 /*定义变量 z*/
    if (x>y) z=x;          /*如果 x>y , 将 x 赋值给 z*/
    else z=y;              /*如果 x<y , 将 y 赋值给 z */
    return (z);           /*把 z 的值返回给函数 max*/
}

```

程序运行情况：

8.5 ✓ (输入 8 和 5 赋给 a 和 b)

max=8 (输出 c 的值)

说明：本程序包括主函数 main 和被调用函数 max 两个函数。max 函数的作用是将 x 和 y 中较大者的值赋给变量 z。return 语句将 z 的值返回给主调函数 main。

1.4.2 C 程序构成简介

(1) C 程序是由函数构成的。这使得程序容易实现模块化。

(2) 一个函数由两部分组成，即函数的首部和函数体。

函数的首部：【例 1-6】中的 max 函数首部为

```
int max(int x,int y)
```

函数体：花括号内的部分。若一个函数有多个花括号,则最外层的一对花括号为函数体的范围。函数体又包括两部分，即声明部分和执行部分。

如例 1-6 中的声明部分 int a,b,c; 可缺省。

执行部分：由若干个语句组成。可缺省。

例如：

```
void dump ( )
{
}

```

这是一个空函数,什么也不做,但是合法的函数。

(3) C 程序总是从 main 函数开始执行的,与 main 函数的位置无关。

(4) C 程序书写格式自由,一行内可以写几个语句,一个语句可以分写在多行上,C 程序没有行号。

(5) 每个语句和数据声明的最后必须有一个分号。

(6) C 语言本身没有输入输出语句, 输入和输出的操作是由库函数 `scanf` 和 `printf` 等函数来完成的。C 对输入输出实行“函数化”。

1.5 C 语言程序的执行

1.5.1 C 程序的运行步骤

按照 C 语言语法规则编写的 C 程序称为源程序。事实上, 计算机只能识别和执行由 0 和 1 组成的二进制机器语言, 而不能识别和执行高级语言。为了使计算机能够执行 C 源程序, 设计好 C 源程序后, 必须使用编译软件将源程序翻译成二进制的目标程序, 然后将目标程序和系统的函数库以及其他目标程序链接起来, 形成可执行的目标程序。所以在编好一个 C 源程序后, 要经过以下几个步骤才能上机运行: 输入编辑源程序→编译源程序→链接库函数→运行目标程序, 具体过程如图 1-15 所示, 其中实线表示操作流程, 虚线表示文件的输入输出。

1. 编辑源程序

设计好的源程序要利用程序编辑器输入计算机, 输入的程序一般以文本文件的形式存放在磁盘上, 文件的扩展名为 `.c`。所用的编辑器可以是任何一种文本编辑软件, 比如像 Turbo C 和 VC++ 这样的专用编辑系统, 或者是 Windows 系统提供的写字板或字处理软件等都可以用来编辑源程序。

2. 编译源程序

源程序是无法直接被计算机执行的, 因为计算机只能执行二进制的机器指令, 这就需把源程序先翻译成机器指令, 然后计算机才能执行翻译好的程序, 这个过程是由 C 语言的编译系统完成的。源程序编译之后生成的机器指令程序叫目标程序, 其扩展名为 `.obj`。

3. 链接程序

在源程序中, 输入输出等标准函数不是用户自己编写的, 而是直接调用系统函数库中的库函数。因此, 必须把目标程序与库函数进行链接, 才能生成扩展名为 `.exe` 的可执行文件。

4. 运行程序

执行 `.exe` 文件, 得到最终结果。

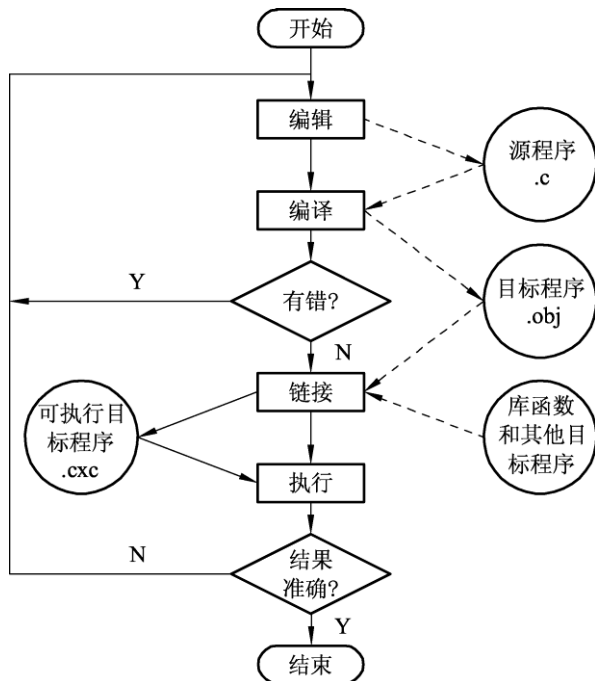


图 1-15 C 程序运行步骤

在编译、链接和运行程序的过程中,都有可能出现问题,此时可根据系统给出的错误提示对源程序进行修改,并重复以上环节,直到得出正确的结果为止。

1.5.2 C 程序的集成开发工具

程序的集成开发工具是一个经过整合的软件系统,它将编辑器、编译器、链接器和其他软件单元集合在一起,在这个工具里,程序员可以很方便地对程序进行编辑、编译、链接以及调试程序的运行过程,以便发现程序中的问题。

C 程序的集成开发工具很多,如 Turbo C、Microsoft C、Visual C++、Dev C++、Borland C++、C++ Builder、Gcc、CodeBlocks、Visual Studio、Clion、Visual Studio Code 等。这些集成开发工具各有特点,分别适用于 DOS 环境、Windows 环境和 Linux 环境,几种常用的 C 程序开发工具的基本特点和所适用的环境如表 1-2 所示。

表 1-2 C 程序的集成开发工具表

开发工具	运行环境	各工具的差异	基本特点
Turbo C	DOS	不能开发 C++ 语言程序	(1) 符合标准 C; (2) 各系统具有一些扩充内容; (3) 能开发 C 语言程序(集程序编辑、编译、链接、测试、运行于一体)
Microsoft C	DOS		
Visual C++	Windows	能开发 C++ 语言程序 (集程序编辑、编译、链接、测试、运行于一体)	
Dev C++	Windows		
Borland C++	DOS、Windows		
C++ Builder	Windows		
Gcc	Linux		
CodeBlocks	Windows、Linux		
Visual Studio	Windows、Linux		
Clion	Windows、Linux		
Visual Studio Code	Windows、Linux	(1) 具有强大的代码补全功能; (2) 支持语法高亮	

从表 1-2 可以看出,有些集成开发工具不仅适合开发 C 语言程序,还适合开发 C++ 语言程序。这些既适合 C 语言又适合 C++ 语言的开发工具,一开始并不是为 C 语言而写的,而是为 C++ 语言设计的集成开发工具,但是因为 C++ 语言是建立在 C 语言基础之上的,C 语言的基本表达式、基本结构和基本语法等方面同样适合 C++ 语言,因此,这些集成开发工具也能开发 C 语言程序。

Dev C++ 是一个 Windows 下的 C 和 C++ 程序的集成开发环境。它使用 MingW32/GCC 编译器,遵循 C/C++ 标准。开发环境包括多页面窗口、工程编辑器以及调试器等,在工程编辑器中集合了编辑器、编译器、连接程序和执行程序,提供高亮度语法显示,以减少编辑错误,还有完善的调试功能,能够适合初学者与编程高手的不同需求,是学习 C 或 C++ 的首选开发工具。

VC++ 是 Microsoft 公司以 C++ 为基础开发的可视化集成开发工具。Microsoft Visual C++ 2010 版本,是微软公司于 1998 年 6 月 29 日发布的,是目前世界上最流行的 C++ 开发工具,

同时也是 Microsoft Visual Studio(tm) 6.0 开发系统的成员之一。Visual C++ 2010 为不断增长的 C++ 开发产业带来了一系列提高生产力的新功能，这些新功能能够在不牺牲 Visual C++ 所特有的强大功能与性能的同时，提高程序的编写速度。另外，Visual C++ 2010 还将提供更好的对 Web 与企业开发的支持。Visual C++ 中加入的 IntelliSense(r) 技术能够使开发人员编写代码的工作变得更快捷和更容易，新的“Edit 和 Continue”调试功能能够使开发人员做到以前完全不可能做到的事情，即不离开调试器就可以对代码进行编辑，从而大大缩短了程序的开发时间。

Turbo C2.0 不仅是一个快捷、高效的编译程序，同时还有一个易学、易用的集成开发环境。使用 Turbo C2.0 无须独立地编辑、编译和连接程序，就能建立并运行 C 语言程序。因为这些功能都组合在 Turbo 2.0 的集成开发环境内，并且可以通过一个简单的主屏幕使用这些功能。

1.6 小 结

程序设计语言分为机器语言、汇编语言、高级语言三类。程序设计方法主要有结构化设计方法和面向对象设计方法两种，结构化程序设计方法的主要观点是采用自顶向下、逐步求精及模块化的程序设计方法；使用三种基本控制结构构造程序，任何程序都可用顺序、选择、循环三种基本控制结构构造。结构化程序设计主要强调的是程序的易读性。同时，养成良好的程序设计风格是初学者必须要认真关注的事。

C 语言是一种结构化程序设计语言，产生至今已逾 50 年，却仍然在编程软件排行榜稳居 1、2 名，尤其是近年来嵌入式软件的需求量呈爆发式增长，使得 C 语言的优势经久不衰。作为一个经典的程序设计语言，C 语言也不多地修订更新，其标准除了经典的 C89、C99，最新的是 2011 年的 C11 标准。

C 语言既具有高级语言的特点，又具有汇编语言的特点。C 语言提供了许多低级处理的功能，但仍然保持着良好的跨平台特性，以一个标准规格写出的 C 语言程序可在许多计算机平台上进行编译。

C 语言的运行必须通过主函数 main 实现，其运行步骤分为编辑、编译、链接、运行 4 步，其集成开发工具种类繁多。

1.7 习 题

一、选择题

1. C 语言是一种 ()。
A. 机器语言 B. 汇编语言 C. 高级语言 D. 低级语言
2. 下列各项中，不是 C 语言的特点是 ()。
A. 语言简洁、紧凑，使用方便 B. 数据类型丰富，可移植性好
C. 能实现汇编语言的大部分功能 D. 有较强的网络操作功能
3. 下列叙述正确的是 ()。
A. C 语言源程序可以直接在 DOS 环境中运行

- B. 编译 C 语言源程序得到的目标程序可以直接在 DOS 环境中运行
 - C. C 语言源程序经过编译、连接得到的可执行程序可以直接在 DOS 环境中运行
 - D. Turbo C 系统不提供编译和连接 C 程序的功能
4. 下列叙述错误的是 ()。
- A. C 程序中的每条语句都用一个分号作为结束符
 - B. C 程序中的每条命令都用一个分号作为结束符
 - C. C 程序中的变量必须先定义, 后使用
 - D. C 语言以小写字母作为基本书写形式, 并且 C 语言要区分字母的大小写
5. 一个 C 程序的执行是从 ()。
- A. 本程序的 main 函数开始, 到 main 函数结束
 - B. 本程序文件的第一个函数开始, 到本程序文件的最后一个函数结束
 - C. 本程序文件的第一个函数开始, 到本程序 main 函数结束
 - D. 本程序的 main 函数开始, 到本程序文件的最后一个函数结束

二、填空题

1. 汇编语言属于面向 () 的语言, 高级语言属于 () 的语言。
2. 用高级语言编写的程序称为 () 程序, 它可以通过解释程序翻译一句执行一句的方式执行, 也可以通过编译程序一次翻译产生 () 程序, 然后执行。
3. 各类计算机语言的发展历程大致为: 先有 () 语言, 再有汇编语言, 最后出现中级语言和 () 语言。

三、简答题

1. 简述 C 语言的主要特点。
2. C 语言的主要应用领域是什么?
3. 写出一个 C 语言的构成。